

```
1 package org.junit.jupiter.my_maven_project;
2
3 public class Fraction {
4     private int zaehler;
5     private int nenner;
6
7     // icomplete
8     public Fraction(int zaehler, int nenner) {
9         this.zaehler = zaehler;
10        this.nenner = nenner;
11    }
12
13    // wrong
14    public void Kehrwert() {
15        this.setNenner(this.zaehler);
16        this.setZaehler(this.nenner);
17    }
18
19    // wrong
20    public void multiplikation(Fraction a) {
21        this.setNenner(this.nenner * nenner);
22        this.setZaehler(this.zaehler * zaehler);
23    }
24
25    // wrong
26    public void addition(Fraction a) {
27        if (a.getNenner() != this.getNenner()) {
28            int neuerZaehler;
29            this.setZaehler(this.getZaehler() * a.getNenner());
30            neuerZaehler = a.getZaehler() * this.getNenner();
31            this.setZaehler(this.getZaehler() + neuerZaehler);
32        } else {
33            this.setZaehler(this.getZaehler() + a.getZaehler());
34        }
35    }
36
37    // wrong
38    public void kuerzen() {
39        int ggt = ggt(Math.abs(zaehler), Math.abs(nenner));
40        this.zaehler /= ggt;
41        this.nenner /= ggt;
42    }
43
44    // helper method Euklidian algorithm,
45    // here are no mistakes
46    private int ggt(int a, int b) {
47        while (b != 0) {
48            int temp = b;
49            b = a % b;
50            a = temp;
51        }
52        return a;
53    }
54
55    @Override
56    public String toString() {
57        return this.getZaehler() + "/" + this.getNenner();
58    }
59
60    public int getZaehler() {
61        return zaehler;
62    }
63
64    public void setZaehler(int zaehler) {
65        this.zaehler = zaehler;
66    }
67
68    public int getNenner() {
69        return nenner;
70    }
71
72    public void setNenner(int nenner) {
73        this.nenner = nenner;
74    }
75 }
76
```

```
1 package org.junit.jupiter.my_maven_project;
2
3 import my_maven_project.src.test.java.junit.Fraction;
4
5 import org.junit.jupiter.api.Test;
6 import static org.junit.jupiter.api.Assertions.assertThrows;
7 import static org.junit.jupiter.api.Assertions.assertEquals;
8
9 public class TestFraction {
10     @Test
11     public void constructor() {
12         try {
13             Fraction b = new Fraction(1, 0); // This will throw an exception
14         } catch (IllegalArgumentException e) {
15             System.out.println("Fehler: " + e.getMessage()); // Handle the exception
16         }
17         assertThrows(IllegalArgumentException.class, () -> new Fraction(1, 0), "Es kann nicht durch 0 geteilt
werden!");
18     }
19
20     @Test
21     public void additionGleicherNenner() {
22         Fraction a = new Fraction(1,2);
23         Fraction b = new Fraction(2,2);
24         a.addition(b);
25         assertEquals(3, a.getZaehler(), "Zaehler nicht richtig berechnet!");
26         assertEquals(2, a.getNenner(), "Nenner nicht richtig berechnet!");
27     }
28
29     @Test
30     public void additionVerschiedenerNenner() {
31         Fraction a = new Fraction(1,2);
32         Fraction b = new Fraction(1,3);
33         a.addition(b);
34         // 1/2 + 1/3 = 3/6 + 2/6 = 5/6
35         assertEquals(5, a.getZaehler(), "Zaehler nicht richtig berechnet!");
36         assertEquals(6, a.getNenner(), "Nenner nicht richtig berechnet!");
37     }
38
39     @Test
40     public void kehrwertTest() {
41         Fraction b = new Fraction(5,6);
42         b.kehrwert();
43         assertEquals(6, b.getZaehler(), "Zaehler nicht richtig berechnet!");
44         assertEquals(5, b.getNenner(), "Nenner nicht richtig berechnet!");
45     }
46
47     @Test
48     public void multiplikationTest() {
49         Fraction a = new Fraction(2, 5);
50         Fraction b = new Fraction(2, 3);
51         b.multiplikation(a);
52         assertEquals(4, b.getZaehler(), "Zaehler nicht richtig berechnet!");
53         assertEquals(15, b.getNenner(), "Nenner nicht richtig berechnet!");
54     }
55
56     @Test
57     public void kuerzenTest() {
58         Fraction b = new Fraction(2,4);
59         b.kuerzen();
60         assertEquals(1, b.getZaehler(), "Falsch gekuerzt !");
61         assertEquals(2, b.getNenner(), "Falsch gekuerzt ");
62     }
63 }
64
```

- my-maven-project
 - src/main/java
 - org.junit.jupiter.my_maven_project
 - Bruch.java
 - Fraction.java
 - src/test/java
 - org.junit.jupiter.my_maven_project
 - TestBruch.java
 - TestFraction.java
 - JRE System Library [JavaSE-21]
 - Maven Dependencies
 - src
 - target
 - pom.xml

```
21 public void addNenner(Nenner n) {  
22     Fraction a = new Fraction(1,2);  
23     Fraction b = new Fraction(2,2);  
24     a.addition(b);  
25 }
```

Tasks Terminal Console Git Staging JUnit x PlantUML

Finished after 0.061 seconds

Runs: 6/6

Errors: 1

Failures: 4

TestFraction [Runner: JUnit 5] (0.020 s)

additionVerschidenerNenner() (0.014 s)

kuerzenTest() (0.001 s)

multiplikationTest() (0.001 s)

constructor() (0.001 s)

kehrwertTest() (0.000 s)

additionGleicherNenner() (0.000 s)

Failure Trace

org.opentest4j.AssertionFailedError: Nenner nicht richtig berechnet! \Rightarrow expected: <6> but was: <2>

at org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:151)

at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)

at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)

```
1 package org.junit.jupiter.my_maven_project;
2
3 public class Bruch {
4     private int zaehler;
5     private int nenner;
6
7     // incomplete
8     public Bruch(int zaehler, int nenner) {
9         this.zaehler = zaehler;
10        this.nenner = nenner;
11        if (nenner == 0) throw new IllegalArgumentException("Nenner darf nicht 0 sein!");
12    }
13
14    // wrong
15    public void kehrwert() {
16        int n = this.nenner;
17        this.setNenner(this.zaehler);
18        this.setZaehler(n);
19    }
20
21    // wrong
22    public void multiplikation(Bruch a) {
23        this.setNenner(this.nenner * a.getNenner());
24        this.setZaehler(this.zaehler * a.getZaehler());
25    }
26
27    // wrong
28    public void addition(Bruch a) {
29        if (a.getNenner() != this.getNenner()) {
30            int neuerZaehler;
31            int neuerNenner = this.getNenner() * a.getNenner();
32            this.setZaehler(this.getZaehler() * a.getNenner());
33            neuerZaehler = a.getZaehler() * this.getNenner();
34            this.setZaehler(this.getZaehler() + neuerZaehler);
35            this.setNenner(neuerNenner);
36        } else {
37            this.setZaehler(this.getZaehler() + a.getZaehler());
38        }
39    }
40
41    // wrong
42    public void kuerzen() {
43        int ggt = ggt(Math.abs(zaehler), Math.abs(nenner));
44        this.zaehler /= ggt;
45        this.nenner /= ggt;
46    }
47
48    // helper method Euklidian algorithm,
49    // here are no mistakes
50    private int ggt(int a, int b) {
51        while (b != 0) {
52            int temp = b;
53            b = a % b;
54            a = temp;
55        }
56        return a;
57    }
58
59    @Override
60    public String toString() {
61        return this.getZaehler() + "/" + this.getNenner();
62    }
63
64    public int getZaehler() {
65        return zaehler;
66    }
67
68    public void setZaehler(int zaehler) {
69        this.zaehler = zaehler;
70    }
71
72    public int getNenner() {
73        return nenner;
74    }
75
76    public void setNenner(int nenner) {
77        this.nenner = nenner;
78    }
79 }
80
```

```
1 package org.junit.jupiter.my_maven_project;
2
3 import my_maven_project.src.test.java.junit.Bruch;
4
5 import org.junit.jupiter.api.Test;
6 import static org.junit.jupiter.api.Assertions.assertThrows;
7 import static org.junit.jupiter.api.Assertions.assertEquals;
8
9 public class TestBruch {
10     @Test
11     public void constructor() {
12         try {
13             Bruch bruch = new Bruch(1, 0); // This will throw an exception
14         } catch (IllegalArgumentException e) {
15             System.out.println("Fehler: " + e.getMessage()); // Handle the exception
16         }
17         assertThrows(IllegalArgumentException.class, () -> new Bruch(1, 0), "Es kann nicht durch 0 geteilt
werden!");
18     }
19
20     @Test
21     public void additionGleicherNenner() {
22         Bruch a = new Bruch(1,2);
23         Bruch b = new Bruch(2,2);
24         a.addition(b);
25         assertEquals(3, a.getZaehler(), "Zaehler nicht richtig berechnet!");
26         assertEquals(2, a.getNenner(), "Nenner nicht richtig berechnet!");
27     }
28
29     @Test
30     public void additionVerschiedenerNenner() {
31         Bruch a = new Bruch(1,2);
32         Bruch b = new Bruch(1,3);
33         a.addition(b);
34         // 1/2 + 1/3 = 3/6 + 2/6 = 5/6
35         assertEquals(5, a.getZaehler(), "Zaehler nicht richtig berechnet!");
36         assertEquals(6, a.getNenner(), "Nenner nicht richtig berechnet!");
37     }
38
39     @Test
40     public void kehrwertTest() {
41         Bruch b = new Bruch(5,6);
42         b.kehrwert();
43         assertEquals(6, b.getZaehler(), "Zaehler nicht richtig berechnet!");
44         assertEquals(5, b.getNenner(), "Nenner nicht richtig berechnet!");
45     }
46
47     @Test
48     public void multiplikationTest() {
49         Bruch a = new Bruch(2, 5);
50         Bruch b = new Bruch(2, 3);
51         b.multiplikation(a);
52         assertEquals(4, b.getZaehler(), "Zaehler nicht richtig berechnet!");
53         assertEquals(15, b.getNenner(), "Nenner nicht richtig berechnet!");
54     }
55
56     @Test
57     public void kuerzenTest() {
58         Bruch b = new Bruch(2,4);
59         b.kuerzen();
60         assertEquals(1, b.getZaehler(), "Falsch gekuerzt !");
61         assertEquals(2, b.getNenner(), "Falsch gekuerzt ");
62     }
63 }
64
```

- my-maven-project
 - src/main/java
 - org.junit.jupiter.my_maven_project
 - Bruch.java
 - Fraction.java
 - src/test/java
 - org.junit.jupiter.my_maven_project
 - TestBruch.java
 - TestFraction.java
 - JRE System Library [JavaSE-21]
 - Maven Dependencies
 - src
 - target
 - pom.xml

```
21 public void additonGleichenNenner() {  
22     Bruch a = new Bruch(1,2);  
23     Bruch b = new Bruch(2,2);  
24     a.addition(b);  
...
```

Tasks Terminal Console Git Staging JUnit x PlantUML

Finished after 0.062 seconds

Runs: 6/6 Errors: 0 Failures: 0

TestBruch [Runner: JUnit 5] (0.015 s)

- additionVerschidenerNenner() (0.010 s)
- kuerzenTest() (0.000 s)
- multiplikationTest() (0.001 s)
- constructor() (0.001 s)
- kehrwertTest() (0.000 s)
- additionGleicherNenner() (0.000 s)

Failure Trace